

DOCUMENT RESUME

ED 328 220

IR 014 816

AUTHOR O'Malley, Claire
TITLE Interface Issues for Guided Discovery Learning
Environments. CITE Report No. 85.
INSTITUTION Open Univ., Walton, Bletchley, Bucks (England). Inst.
of Educational Technology.
PUB DATE Aug 89
NOTE 40p.
PUB TYPE Information Analyses (070) -- Viewpoints (120)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Computer Assisted Instruction; *Computer System
Design; Discovery Learning; Educational Environment;
*Expert Systems; Foreign Countries; *Interaction;
*Man Machine Systems; Pictorial Stimuli; Programed
Tutoring

ABSTRACT

The nature of the learner-machine interaction is of central importance in the design of guided discovery learning environments. The interface to a learning environment is the only way in which a learner has access to the functionality of the system, and conversely, the interface is the only way in which the tutoring component of the system has access to the activities of the learner. Ideally, the interface system should be designed so that the system itself becomes the domain to be learned, rather than a representation of that domain. However, such a "direct manipulation interface" is practically impossible given the infinite number of ways that learners interact with the domain to be learned, with the symbols the interface uses to represent that domain (e.g., pictorial or iconic stimuli), and with the tasks necessary to manipulate those symbols within the interface. Similar complexities arise when determining how the learner is represented to the system, or the extent to which the interface provides a source of evidence for the tutoring system to make inferences about the learner. Such a learner-machine interface must take into account the learner's prior understanding, and the context in which the interaction takes place. (66 references) (DB)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

- ☐ This document has been reproduced as received from the person or organization originating it.
- ☐ Minor changes have been made to improve reproduction quality.
- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Interface issues for guided discovery learning environments

Clair O'Malley

CITE Report No. 83

"PERMISSION TO REPRODUCE THIS
MATERIAL HAS BEEN GRANTED BY

E.S. Henderson

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

Interface issues for guided discovery learning environments[†]

Claire O'Malley
Institute of Educational Technology
The Open University
Walton Hall
Milton Keynes
MK7 6AA

1. Introduction

This chapter takes up one of the themes outlined by Elsom-Cook (in press) — that the nature of the learner-machine interaction is of central importance in the design of guided discovery learning environments. Two aspects of this are explored: the first is that the interface to a learning environment (whether it's "non-intelligent" CAL or "intelligent" tutoring systems) is the only way in which the learner has access to the functionality of the system. The implication is that we can no longer afford to design the nuts and bolts of the system and simply "tack on" an interface afterwards. The underlying functionality of the system (not the particular implementation, but the actual functional behaviour) cannot be completely separated from the user interface. Functionality "leaks through" into interaction. The development of the learner's model of the system and the domain being taught are shaped by what the learner-as-user can see, hear, do with the interface.

This chapter focuses on the design of iconic interfaces in particular. It is argued that careful attention needs to be paid to three types of representation at the interface. The first is the representation of the semantics of the domain to be learned. The second is the representation of the system's functionality. This refers to the actual operations the system performs in representing the domain. It is treated as a separate type of representation, since it may not map completely onto the domain semantics. The third type of representation concerns what the user is expected to do in interacting with the system. This can be regarded as the "affordance" which the system offers for interaction.

The other aspect of the learner-machine interface that is explored is the mapping from the user's actions to the interpretation of those actions in a pedagogical context. It is argued that the interface is also the only way in

[†] To appear in M.T. Elsom-Cook (Ed.) Guided Discovery Tutoring, Paul Chapman Publishers, London.

Thanks to Mark Elsom-Cook, Rod Moyse and Tim O'Shea for helpful comments in preparing this chapter.

which the tutoring component has access to the activities of the learner. The implication of this is that the tutoring or guidance system must have some representation of the learner-machine interaction itself in order to model the development of the learner's understanding. This applies as much to active diagnosis (assessment) as it does to passive diagnosis (modelling behaviour). The issue of what needs to be represented even for a sufficiently accurate model (as opposed to an ideal model) is problematic: the learner-machine interface is not bounded by the computer screen alone but includes the physical and social context in which the learner-machine system is embedded.

2. The role of the interface in educational environments

2.1 Interface as knowledge representation

The design of the user interface to a piece of educational software is of vital importance to its educational effectiveness. This applies whether one is designing intelligent tutoring systems, microworlds, simulations or any other form of computer-based learning activity. However, as Frye and Soloway (1987) have pointed out, it is an issue that is largely neglected in the field of information technology in education.

The importance of the interface in learning environments and intelligent tutoring systems is also emphasised by Wenger (1987), who distinguishes between two sources of knowledge representation in tutoring systems: an internal representation of the domain, and an external representation at the interface. Wenger argues that the interface does more than map meaning onto a set of symbols; the external representation can actually compete with the internal representation as a source of domain knowledge. He also points out that the external representation does not always play the ancillary role traditionally associated with user interfaces, but can be the driving force of the design of a tutoring system. For example, the design of Guidon (Clancey, 1983) took the traditional path from the construction of an internal model (domain representation), to augmenting the system with visual displays in Guidon-Watch (Richer and Clancey, 1985), whereas the design of Steamer (Hollan, Hutchins and Weitzman, 1984) went from an external representation (a graphical interface) to an internal representation (using qualitative process theory), which was specifically designed to incorporate the conceptual content of the visual display into the internal representation.

It is argued in this chapter that environments for guided discovery learning need to be designed "outside-in" rather than "inside-out". There are two main reasons for this. The first is to ensure that the resulting design is usable to the extent that the interface "disappears" for the user so that he or she concentrates on the domain itself. The second is to ensure that the representation by the tutoring system of the learner-as-user is

sufficiently accurate. This accords with John Seely Brown's analysis of the requirements for producing "glass-box technology" (Brown, 1989). Brown argues that systems need to be made "transparent" in three senses: *domain* transparency, allowing the user to see "through" the tool; *internal* transparency allowing the user to see "into" the tool; *embedding* transparency, allowing the user to see the relationship of the technology to the larger context of the interaction. The analysis in this chapter elaborates on these three aspects under slightly different headings. Firstly, domain transparency and internal transparency are discussed with respect to how the system represents itself to the user via the interface. Iconic interfaces are taken as exemplars for an analysis of how these two types of transparency might be achieved. The notion of embedding transparency is examined with respect to how the user is represented to the tutoring system. It is argued that the guidance component of any learning environment requires an explicit representation of the "outside" or context of the learner-machine interaction.

2.2 Making the interface transparent

If people are using computers to learn about something their task should not be made more difficult by having to struggle with learning and using the computer system itself. Few would argue with the suggestion that educational interfaces should be easily learnable and usable, but the argument is a little more subtle than this. If the learner has to concentrate on using the interface, then attention is drawn away from the domain being taught.

The interface is the only way in which the user can operate in the represented world. For the purposes of learning, the interface should in some sense *be* the represented world — the domain to be learned. If it is designed appropriately the interface becomes transparent and no longer exists for the user. The represented world becomes, cognitively, directly present. This quality of being directly engaged in the domain is captured to some extent in the notion of "direct manipulation" interfaces.

There are two ways to think about direct manipulation. One is as a descriptor for a class of interfaces, usually characterised as being graphical, rather than text-based, and involving the use of pointing devices (e.g., mice) for physically manipulating graphical objects on the screen. An example in the educational domain is *Steamer* — a simulation environment for learning about steam plants (Hollan, Hutchins & Weitzman, 1984). Shneiderman (1983) describes direct manipulation interfaces as those which have the characteristics of continuous representation of items of interest, rapid incremental reversible operations with immediate visible feedback and direct physical actions (e.g., pointing with a mouse) instead of complex syntax (i.e., constructions in a command language). This description is unsatisfactory for several reasons: the experience of "direct engagement" does not necessarily entail graphical

interaction — indeed, for some aspects of interaction, graphical or iconic interfaces provide a very *indirect* mapping between the user's goals and the tasks they have to perform to achieve those goals. Some information retrieval tasks, for example, are probably much better suited to command-based interactions, particularly where quantifiers are involved in the specification of the search. Simply replacing a command language with physical manipulation and pointing via a mouse or joystick does not guarantee the avoidance of errors or problems in learning on the part of the user. The physical actions involved in using a mouse or any other pointing device have syntax as well as the expressions in a command language entered via the keyboard.

A second way to think about direct manipulation is as a theory of interaction. The beginning of such a theory has been offered by Hutchins et al. (Hutchins, Hollan and Norman, 1986). The starting point for their analysis is to examine the gap that exists between the user's goals and system tasks on the one hand (the "gulf of execution") and the gap between system output and the user's goals (the corresponding "gulf of evaluation" — see Figure 2.1). Successful interaction is helped by reducing both of these gaps.

These are cognitive gaps and relate to an individual user's goals. What is "direct" for one user, may not be so for another. The factor of directness relates to prior experience that the user has. If the user is an experienced Unix user, the Unix interface can be direct, but not so for the novice. The interface may eventually become direct with experience, but if we're designing interfaces for learning, it is important to minimise the learning of the interface itself. On the other hand, there are some gaps that will exist for all users. For example, a text editor which only has operations for characters and words will be indirect for supporting the task of moving larger sections of text around. The goal of moving a section has to be translated into the goal of moving through the document to the beginning and end of the section, placing marks and issuing a command. The sequence of actions required cannot become automatic with practice but has to be planned each time. The commands themselves (i.e., locate beginning of section, mark beginning of section, locate end of section, mark end of section, delete section, locate new position for section, replace deleted section) can be learned, as can the sequence of commands. This sequence has far more steps involved than simply locating the section and applying a simple cut-and-paste operation. There are more opportunities for the user to make an error at each stage, or for losing track of the sequence of activities.

These points are brought out in the distinction that is made between two kinds of directness: "semantic directness" on the one hand, and "articulatory directness" on the other. Semantic directness refers to the mapping between the user's goals or intentions and the means provided by the system for expressing those intentions. An interface is semantically

indirect if the user has to engage in a good deal of planning and problem solving in using the system to perform some task. Articulatory directness (or referential directness — cf. Hutchins, 1987) refers to the extent to which the user's understanding of the meaning of expressions maps onto the form of expressions required by the system. The distinction is similar in some respects to that drawn between semantics and syntax, but articulatory directness refers not simply to the formal properties of an expression, it includes the pragmatic aspects of interaction. These aspects of interaction cannot be sharply differentiated, however. The syntax of an expression is often closely related to its semantics. For example, a substitution expression in a text editor may take the form *substitute* <new item> <old item>, or it may take the form *substitute* <old item> <new item>. This may or may not map onto the user's understanding of the meaning or use of the natural language equivalent expression "substitute" (in fact, novice users have trouble with this construct in the Unix text editor *vi* — Riley and O'Malley, 1985).

The next section discusses some of the implications of the direct manipulation approach to the design of interfaces for exploratory or discovery learning environments. The notions of semantic and articulatory directness are examined with respect to three different aspects of how the system represents itself to the user. The first is how the system represents the domain — in this case, the domain to be learned. The second is how the system represents its functionality. The third aspect is how the system represents what the user needs to do in the interaction. These aspects are separated here because what was intended in designing the particular domain representation and what is actually presented to the user may not be one and the same.

3. (Re)presenting the system to the user

Elsom-Cook (in press) argues that, in order to provide guidance in discovery learning environments, the system needs to model the processes whereby the student learns from interacting with the system. Even if we can develop theories of learning which are detailed enough for student modelling, they could not be applied to specific situations and adapted for particular learners, without being attached to domain semantics. It follows, both from this, and from the previous discussion of the interface as knowledge representation, that we need to develop a theory of interaction. We need to understand how learning takes place via interaction with the system.

3.1 Representing domain semantics

In order to teach something new, we should build on what students already know (see the discussion of "cognitive hooks" in Chapter 1). Brown and Clement (1987) discuss this in terms of providing "anchors"

rooted in the student's experience, from which to draw analogies with new concepts. One approach is to represent the domain to be taught in the form of a simulation. An example is Steamer (Hollan et al., 1984). Some domains (e.g., steam plants, electrical circuits) are more amenable to simulation than others (e.g., mathematics). Another approach is to design the interface with appropriate metaphors, as in microworlds such as Shopping-on-Mars (Hennessy, Evertsz, Ellis, Black, O'Shea & Floyd, 1987) or the Alternate Reality Kit (Smith, 1986).

There are two main advantages in using metaphors at the interface. One is that judicious use of metaphors allows users to draw analogies with their own experience. This enhances the learnability of the system because the concepts and techniques necessary for using the system are already familiar. Another advantage in using a consistent and uniform metaphor at the interface is that it enables users to make correct predictions about the behaviour of the system, even in novel or unfamiliar circumstances. These rationales lie behind the "desktop" metaphor of the Apple Macintosh, for example, and the "physical world" metaphor of ARK.

ARK (Smith, 1986) is a system for creating interactive animated simulations. It is built on a metaphor of the physical world in which all objects have a visual image, a position and velocity and are operated on by physical forces. The user interacts with this world via a mouse-operated "hand", which they can use to pick up and throw objects, press buttons, operate sliders and so on. The use of the hand contributes a good deal to understanding the system. Users don't have to be told the syntax for interacting with the objects represented in the interface.

Everything in ARK can be manipulated according to the constraints of physical laws, in a faithful simulation of the real world. However, there is a question about the extent to which it is necessary to map representations at the interface onto the real world, or, rather, onto users' experiences of the real world. After all, one of the advantages of using computers in education is that we can use the power of computation to represent aspects of the domain which are difficult to perceive in the real world, or with other kinds of educational technology. Randall Smith talks about this as the "tension between literalism and magic" (Smith, 1987) in interfaces. Literal features are those which are faithful to the interface metaphor, whereas "magical" features deliberately violate the metaphor in order to achieve enhanced functionality. ARK differs from the real world in that the physical laws can be changed. Users can create artificial microworlds within the environment with which to compare the physical laws governing the real world and explore how they interact with one another. Scanlon and Smith (1988) used ARK successfully to introduce users who had little or no physics training to the behaviour of particles in a bubble chamber. ARK has also been used in a series of experiments on students' misconceptions of conservation of momentum (O'Shea & Smith, 1987).

Semantic directness involves mapping from the user's goals and mental models onto the representations at the interface. This certainly accords with a general pedagogical strategy of attaching the learning situation to students' existing knowledge. Whereas it's relatively straightforward in domains such as office tasks to take on board the goal of semantic directness which Hutchins et al. set before us, it is less easy to achieve in educational environments. For example, analytic techniques such as TAKD (Johnson, Diaper and Long, 1985) can be used to analyse the goals and tasks involved in using mail to map onto the design of an appropriate interface for electronic mail, but very often in educational domains there is no clear pre-existing task structure or set of goals to be analysed. In educational environments we usually need to introduce new domains, concepts and goals. Often these are highly abstract notions, such as velocity in teaching physics, which are not represented in any direct way in the real world or in students' experience.

Elsom-Cook (in press) discusses the problems of representing directly abstract concepts such as recursion in direct manipulation environments such as ARK. However, the arguments about providing "cognitive hooks" are not arguments for always representing the domain in terms of real-world experience. That is only an entry point. The idea is that, via careful use of metaphors and a controlled transition between different metaphors, the learner can be introduced to the appropriate abstract representations, which may later be formally represented (e.g., in Lisp code).

It is possible, then, to introduce new concepts without having to sacrifice the learner's prior understanding. However, this may involve a tradeoff between consistency of representation in the interface and consistency with users' prior knowledge. There are two solutions to this: one is to accept that some aspects of the system (i.e., the "magical" features) will have to be explained to the user explicitly; the other is to use different metaphors for different aspects of the domain. Many people have argued that multiple representations of this kind are vital for learning (e.g., Clement, 1988; Collins & Brown, 1988).

However, others have argued that the use of multiple representations needs to be carefully engineered in an "orderly progression" (White & Frederiksen, 1987; Scanlon & O'Shea, 1988), where the relationships between the metaphors are taught explicitly and used to represent some of the abstract notions one is trying to teach. This technique of using "bridging analogies" has been successfully used in teaching physics (Brown & Clement, 1987). The relationship between the metaphors can also be used to isolate the appropriate dimensions of the analogy while discouraging the importation of inappropriate aspects. It can also be used to provide a source of conflict amongst users' expectations, which may be important in learning (O'Malley, 1987; O'Shea, O'Malley & Scanlon, 1988).

This strategy means that we have to abandon the notion of complete consistency *within* the semantics of the domain in favour of consistency with the user's prior understanding. There is, however, some reason to believe that the learner's knowledge is, in any case, fragmentary and distributed in a similar way (Carroll & Mack, 1985; DiSessa, 1986; Shrager, 1987). Understanding how to make use of multiple representations in a principled way offers some challenges to designers of educational environments.

3.2 Representing system functionality

The choice of appropriate metaphors to represent domain semantics (whether those domains are concrete or abstract) is one issue. The way in which the interface (re)presents those metaphors is a separate issue. Ideally, there should be a direct mapping between the two, but the constraints of software and hardware technology often require an extra translation step. We may choose to represent a concept such as file deletion using the metaphor of throwing a document in the wastebasket, as in the Apple Macintosh interface. However, what the user actually sees is that the document icon can be dragged over the wastebasket icon, which then highlights and when the mouse button is released the document icon disappears and the wastebasket bulges. There is therefore a distinction between the choice of a representation for domain semantics and whether or not that representation works when it is implemented in a particular device.

The design of metaphors such as the "desktop" is usually achieved via graphical or iconic means. Command-based systems which use textual interaction can also convey metaphors. Moreover, the visual modality is not the only source for presenting a metaphor. Auditory icons, such as the sound of breaking glass to convey deletion of a file can also be powerful sources of metaphor (Gaver, 1986, 1989).

There is little in the way of guidelines from HCI for designing iconic interfaces, although some recent research has begun to address the issue (e.g., Gittins, 1986; Lansdale, 1988; Rogers, 1988, 1989). The trouble is that guidelines are often developed from isolated pieces of experimental work and are usually difficult to apply to the much more complex problem of interacting with an iconic interface which has a rich and diverse functionality.

Icons do not simply represent (refer to) domain semantics, they also represent to a greater or lesser degree the functionality of the underlying system. For example, an icon representing a folder in the Macintosh interface also suggests (via the user's understanding of the semantics being represented, which may be more or less successful) that items such as documents can be placed in or removed from that folder.

Successful interpretation of icons depends upon the types of representation used in mapping from symbol to referent, such as whether it's an abstract or metaphorical relationship. But the icons that go to make up metaphors such as the desktop involve sets of symbols that are related in some way, so interpretation may also depend on mappings between icons. Rogers (1988) suggests three interacting factors which contribute towards effective design of iconic interfaces: the directness of the visual link between an icon and its referent; the extent to which the referents can be differentiated within a set of icons; the degree of compatibility between the relations of graphic elements and the common characteristics of the referents (see Figure 2.2).

Context is even more important for icon design than for verbal commands because there are no rules for mapping between them or inferring relationships between them. However, the fact that they are used in an interface can reduce their possible interpretation. For example, the icons appearing in the tools menu shown in Figure 2.3 may be ambiguous unless the user knows that MacDraw is an application for drawing graphics such as circles, rectangles, etc. In other words, the functionality of the system can narrow the interpretation. But this implies that the user already knows the functionality involved. For novice users, and for users of discovery learning environments, the issue is how to represent that functionality so that it is either "obvious" or at least easily learned.

There have been several representational schemes offered for characterising the mapping between the form of an icon and its underlying referent. Gaver (1986) distinguishes between three types of icon-mapping: symbolic mapping is arbitrary, relying on social convention; nomic mapping is where the icon literally stands for its referent (e.g., a picture of a pair of scissors to denote a pair of scissors); metaphorical mapping makes use of the similarities or analogies between the things to be represented and the representing system — this can involve structural mapping, where structural relations are mapped onto one another, or metonymic mapping, where a feature is used to indicate a whole (see Figure 2.4). Rogers (1989) adopts a different classification scheme, and distinguishes between concrete and abstract icons, each of which can have different types of mappings between icon and referent: resemblance, exemplar, symbolic and arbitrary. Resemblance relations represent analogies with the referent of the icon (e.g., the road sign for falling rocks). Exemplar icons show typical examples of the referent (e.g., a knife and fork to depict restaurant services on motorway signs). Symbolic relations are where the icon depicts a concept at a higher level of abstraction than the image itself (e.g., the use of a glass on the side of a box to indicate fragility). These are distinguished from arbitrary relations where the mapping between icon and referent has to be learned (e.g., the symbol for "no entry" in Figure 2.4).

These categories are not independent. Affective factors may also play a role in interpretation of symbols as well as learned cultural conventions. For example, angularity is often associated with hardness, curved horizontal upward strokes with gaiety, curved downward strokes with sadness, and so on. Analogy can also be used in designing symbolic icons — in particular, in interpreting more abstract concepts. For example, in diagrams of electronic circuits a capacitor is shown by two parallel lines perpendicular to the path of current flow. The use of parallel lines to depict the capacitor shows that its function is to block, or contain, the flow of current, by transforming the current to a difference in potential between two plates.

Icon mappings are easier to learn if they show concrete objects operated on (e.g., a pictorial representation of a file), particularly when the referent is an abstract concept (Hemenway, 1982; Jones, 1983). Icons which depict their referents via arbitrary associations will be least easy to learn, with metaphorical mappings in between (Gaver, 1986). However, in practice, icons may contain a mixture of types of sign-referent relationships. The issue of icon-referent mapping is even more complex with sets of icons that are meant to be related. Rogers (1988) points out the importance of the extent to which relations among graphic elements map onto relations (contrasts and similarities) among the referent set (e.g., using stick figures of different sizes to represent the relationships between baby, woman and man). The advantage of this form of representation is that it can provide the user with visual cues about the way in which the particular domain is structured. Having such knowledge may be important in facilitating learning about the domain. If these relations can be communicated at the interface it gives the user a collective model with which to infer the structure of the system. Gittins (1986) has suggested that a further advantage is that it can make the user explore the visual relationship and organisation of the objects within a system, and that such an organisation will have a facilitative effect on retention. Clearly, icon-referent mapping is an important issue in educational applications.

The issue of icon mapping is complicated even further in interfaces because icons do not simply represent objects or concepts, but also functions, actions and states. There isn't much experimental evidence concerning the representation of actions, but some research (e.g., Szlichcinski, 1980) suggests that people may have prototypical representations of actions and states. Rogers' (1988) experiments on the comprehensibility of icons representing action verbs suggest that there are stereotypic images which are readily associated with different types of actions, but that it depends on the imageability of the action verb (that is, how easily people can visualise or imagine the action). For more abstract verbs, there doesn't appear to be one commonly associated image.

The type of command operation or function being represented is another factor in the comprehensibility of icons. Operations concerned

with processing functions (e.g., print) seem to be best represented as prototypical objects performing this process, but commands requiring manipulative actions (e.g., insert) are best represented by abstract symbols representing a prototypical action sequence (Rohr and Keppel, 1984).

Rogers (1988) found that abstract symbols were judged to be most similar in meaning to relatively concrete types of operation, whilst highly pictorial icons depicting concrete objects with concrete analogies associated with them were considered to be the most similar to the more abstract type of commands. However, she also found that the extent of icon-mapping directness depended on whether it was considered in isolation or as part of a set. When the icon-command mappings were evaluated for a whole command set, rather than individually, it was found that the meaningfulness of abstract symbols diminished. The most effective type of mapping was concrete objects associated with actions together with abstract symbols denoting the state of the action. Those which were least effective were the ones which depicted the command operation through a concrete analogy. This suggests that the discriminability of icon-command mappings within a set has an overriding effect on how meaningful icon-mappings are for individual commands. Individual icons need only be meaningful in terms of their similarity to the referent they are representing. However, a set of icons needs to discriminate between the individual referents. Concrete objects plus abstract symbols possess both distinguishable elements and similar features which correspond to the shared and distinctive attributes of the operations within a set.

In summary, given the choice of a particular metaphor for representing domain semantics, whether or not that metaphor succeeds in conveying the domain semantics to the user when implemented in a particular interface depends on several factors. It depends on the type of representation for icon-referent mapping and for mapping icons within sets; it depends on the type of referent being represented (e.g., object, concept, function, action, state); it also depends on the context of use. Given that those particular aspects of representation work, whether or not the metaphor succeeds with respect to learning the domain depends on the appropriateness of the choice of metaphor. The point is that both aspects of representation (the choice of metaphor and its particular implementation) need to be considered. There is also a third aspect of representation that is important: mapping from the system functionality to the actions required of the user.

3.3 Designing for affordance

Most of the research discussed so far has focussed on the design of iconic interfaces to represent domain semantics, system functions and states. Very little attention, however, has been paid to the use of icons to represent what the user is expected to do in order to execute those functions — the use of icons to represent how to interact with the system.

If we take the Apple Macintosh, there are several syntactic expressions for interacting with icons, none of which is explicitly represented at the interface. For example, some operations only require one mouse click, whilst others require two. Some menu items require the user to select an object before executing a menu command, whilst others do not. These operations differ according to context. For example, at the desktop, you have to select a file before choosing the "open" command, whereas within an application such as MacWrite, you can choose the "open" command without having to specify the object or file beforehand. It is not always obvious which operation is appropriate in which context (although the use of "greying out" menu items can indicate non-selectability). The mapping between mouse clicks and the effects of those actions is totally arbitrary. There is no "natural" mapping between pressing a mouse button twice and opening a file, although there is a reasonably natural mapping between pressing a mouse button once and selecting the file (the metaphor is pressing a button to turn something on — the use of highlighting as feedback also reinforces this).

The use of different cursor icons can be useful in indicating various attributes, such as context or mode, but usually this mapping is also fairly arbitrary and may need to be learned. The use of an arrow for pointing is fairly natural, but the insertion point cursor provides a fairly arbitrary mapping between the intent of the user and the system image. The issue becomes more acute for the learner when cursors may represent multiple functions or states at once (cf. Muller, 1988).

The mapping between user input and resulting actions is not simply a matter of syntax (ease of expression), it also depends on semantics (i.e., naturalness of expression) and pragmatics. These "levels" are interdependent. Buxton (1986) argues that the physical operations involved in manipulating various input devices have an important role to play in the "directness" of an interface. Buxton describes three subtasks which are involved in using a pop-up menu: invoking the menu, navigation to select a menu item, making the selection. What ties these three separate physical tasks into one cognitive task is the tension of holding down the mouse button throughout the interaction. Buxton argues that designing the interaction in this way means that syntax and mode errors are virtually impossible to make, since the concluding action (releasing the mouse button) is a natural consequence to the initial action. The tension in the finger also gives constant feedback about which mode the user is in.

There is some evidence that physical operations and proprioceptive feedback may be even more important in designing interfaces for young children. Scaife (1987) argues that the properties of the sensorimotor aspects of interaction must themselves be interiorised by the child before exploration via the interaction can be effective. He cites an experiment by Pailhous and Bullinger (1978) where young children were unable to

correlate the "cause" of moving a camera with the obtained perceptual effects of the changing field of view. These issues do not apply solely to young children learning sensorimotor skills for the first time, they are also likely to apply to adults and to learning in a variety of situations.

Conversely, sensorimotor experiences may also be a source of interference with the new task to be learned. In the domain of physics, for example, misconceptions may derive partly from sensorimotor experiences in the real world (cf. Brown & Clement, 1987). Because they are "deeply compiled", the source of these misconceptions may be hard to uncover — particularly if sensorimotor activity has no part to play in learning with a computer system.

The input devices most commonly used in educational computing are the keyboard and, more recently, mice or joysticks. However, there are a variety of input devices available in the wider market which offer a whole range of functionality (cf. Buxton, 1986; Baecker and Buxton, 1987). The current impoverished situation in educational computing is partly due to factors such as standardisation (particularly since technological innovations are largely driven by the needs of industry rather than education) and partly, one suspects, due to lack of imagination and initiative on the part of designers (fuelled by cost of development for what is seen as a minority market). Things may change in the not-too-distant future. Devices such as the Dataglove and force-feedback joystick (cf. Foley, 1987) are already available outside the research laboratory. One recent example of an innovative educational design for an input device is Noobie (Druin, 1988). Noobie is large furry animal not unlike a Muppet, but it is also a computer for learning about animal behaviour and physiology. Children can clamber all over Noobie and squeeze or hug different parts of the creature to build different types of animals, which can be seen on the screen in Noobie's stomach, and sounds can be heard as the children interact with Noobie.

There are currently practical limitations on the design of educational computing environments and it is often not possible to use the input device that may be most appropriate for the domain. It may be a while (though perhaps not too long) before devices such as force-feedback joysticks or Datagloves are available for schools. There are ways, however, in which representation in other modalities can be used to overcome some of the limitations of input devices.

For example, in ARK all input is via a mouse-operated cursor represented as a "hand". All objects can be picked up and moved or thrown with this hand. In addition, the hand is used to press buttons, drag sliders and activate menus. This multiple functionality is achieved in ARK via the use of a three-button mouse — essentially leading to three modes of interaction, which may depend on the context (i.e., the semantics of the particular object one is interacting with, buttons versus sliders, for

example). The cursor provides feedback in that the "hand" can be in two states: open or closed, depending on whether or not it is "grasping" an object. The mapping from mouse-button to the effects at the interface is sometimes fairly arbitrary, although it is consistent, and needs therefore to be learned. Moreover, if one is paying attention to the screen, it's quite easy to hit the wrong mouse button. The issue is, how could one redesign the interface so that the mapping is obvious without losing all this functionality?

It is possible to use a single-button mouse and maintain most (if not all) of the functionality required in this situation, by mapping the semantics of the object onto the behaviour and appearance of the cursor in a fairly natural way. For example, let's consider the button represented in Figure 2.5a. In the real world, if I have the goal of pressing a button, as my hand is approaching the button I am most likely to extend my index finger in anticipation of pressing. If, however, I wanted to grab the button, I would miss the centre of it with my index finger and place the centre of my palm over the object to be grabbed, in anticipation of then closing my fist around the object.

However, the system has no knowledge of what my hand is doing when I use a single button mouse, except in a very limited way. The system knows whether the mouse button is up or down, and it knows where the cursor is. With this information, it can also know when I enter or leave an object. We can use this simple information to obtain quite complex behaviour. Figure 2.5a shows the hand approaching the button. The "hotspot" (the location of the cursor given as an x,y coordinate) is on the tip of the index finger. When the cursor point enters the centre of the button (Figure 2.5b), the open hand changes to show the index finger pointing, indicating two things: the functionality of the button (i.e., that it can be pressed) and the way in which the user can interact with the button (i.e., by pressing the mouse button — this information is inferrable because the behaviour of the ARK hand maps directly onto the behaviour of the user's hand). If the user then moves the mouse up and outside of the region of the button (Figure 2.6a), the cursor changes back into an open hand, and the "hotspot" is now located in the centre of the palm. In addition, the hand no longer casts a shadow, indicating that it is on the object, rather than suspended above it. If the user presses the mouse button and holds it down while the palm is centred over the object, the cursor changes into a fist to indicate that the object has been "grabbed" (Figures 2.6b and 2.6c). The user can then drag the object around while the button is pressed down. If the button is released, the fist opens again (Figure 2.6d). When the user moves the open hand outside of the region of the button, the hand casts a shadow once again.

The point of this example is to show how careful consideration of fairly "low level" details in a graphical interface can make a big difference to the experience of "directness" which Hutchins et al. (1986) discuss. In addition,

it indicates in a small way how we might start to think about designing the "affordances" of objects represented in the interface.

4. (Re)presenting the user to the system

The discussion so far has been concerned with the extent to which the interface conveys domain semantics, system functionality and "affordances" of the interaction. This three-way distinction is similar in some ways to the "model-view-controller" (MVC) distinction made by Krasner and Pope (1988) and used in the Smalltalk-80 interface. The model refers to the thing being represented; the view is the way of representing the model; the controller is the way of manipulating the representation to affect the model. These are aspects of the ways in which the system represents itself to the user. The issue discussed in this section concerns how the user is represented to the system — in particular, the extent to which the interface provides a source of evidence for the tutoring system to make inferences about the learner.

4.1 Limitations of the direct manipulation approach

Direct manipulation interfaces such as ARK fall into the category of what Hutchins (1987) describes as "model-world interfaces". Hutchins provides a four-way categorisation of styles or modes of interaction: conversational interfaces, declaration interfaces, model-world interfaces and collaborative manipulation interfaces.

Conversational interfaces are those characterised most closely by command-language styles of interaction such as Unix. They are based on a "conversational metaphor" where interaction with the system is via some intermediary language. Input in such interfaces need not necessarily be via typed input. Menu-driven systems can also be conversational in style. Traditional approaches to intelligent tutoring systems embody this conversational style of interaction in the computer-as-tutor paradigm (this includes directive teaching as well as coaching and socratic methods).

However, certain aspects of command-languages or menu designs can make them more direct than these conversational interfaces. Thus, when an experienced Unix user types "dw" (an abbreviation of "delete word") he or she may experience a direct form of interaction. This is the "declaration" style of interface which Hutchins describes. The analogy is with the use of speech acts to perform some action (e.g., "I'm warning you!").

Hutchins discusses the extent to which these interaction styles can be really direct in terms of the possibility of making errors or performing illegal acts. The use of any kind of language, by virtue of its flexibility of

expression, entails the possibility of making errors. The difference in the model-world interaction style is that the model world constrains the set of possible actions so as to render making an error impossible. However, these constraints also reduce the flexibility of expression which declaration interfaces allow, and thereby, in some sense, reduce the "directness" of the interaction. For example, quantification is awkward at best in interfaces such as ARK. There is no way in which to refer to "some".

Hutchins suggests a way in which an interface can capture the best features of declaration and model-world interactions via "collaborative manipulation". This involves giving the system access to the world in which the user operates, to enable meta-level discussion of the domain. Examples of such interaction styles are found in recent extensions to Steamer (Hutchins, 1987). The "display controller" selects displays and display components, based on the importance of the process variables indicated by the display components, which the operator may then reject, indicating that they are not relevant in the current context. The display controller learns about the operator's preferences in the same way it has learned about the system's behaviour, by observation. The display is thus a shared world of action which can be collaboratively manipulated by the user and the display controller. Another example is Designer (Weitzman, 1986), a graphics design editor which can detect violations of graphic design principles and notify the user. The user can then ask the system to demonstrate ways to correct the violations. Of the four styles of interaction described by Hutchins, the collaborative manipulation style is probably most suited to the goals of guided discovery learning systems.

This notion is similar to that proposed by Elsom-Cook (in press). He proposes a "shared world" where all actions and events which occur in that world are equally accessible to the tutoring system and the learner and where each have a common language for referring to the world. However, this in itself is not enough to guarantee shared understanding between the tutoring system and the learner. The problems involved in the system seeing the world "as the student sees it" are discussed in the next section.

4.2 Defining the boundaries of the interface

We only have an implicit understanding of what is meant by "user-computer interface". If one searches the HCI literature, it seems that it means many things to many different people. For some, it seems restricted to the visual display (i.e., what the user sees), for others it includes the hardware (e.g., input devices), for others it includes the social context in which the computer is being used. It is all these, and more. The user-computer interface is probably best considered as a distributed system. It is partly represented in the mind of the user, in the sense that the user constructs a model of system during the course of the interaction, which may or may not accord with what the designer had in mind. It is partly represented in the functionality of the software, partly in the appearance of

the graphical objects on the screen, partly in the documentation accompanying the system, and so on. (Norman, 1986, discusses this distribution of the interface in terms of a three-way distinction between the designer's model, the user's mental model and the system image, including the documentation.) Furthermore, if one considers the fact that users work with computers together in communities (whether they're offices or classrooms), the interface is also represented in the community (cf. Bannon, 1986).

The slipperiness of the concept of user-computer interface increases when one considers the dynamics of user-computer interaction. Just as the interface is distributed over space (if we extend the notion of space to include "cognitive space"), so it is also distributed across time, and it changes over time. This applies both at the "low" level, in the sense that visual displays are dynamic, and at "higher" levels, in the sense that computer systems are appropriated by communities and given particular cultural interpretation.

Decisions about the location of the boundaries of the interaction have crucial implications for the possibility of modelling the student's understanding of the domain. Wenger (1987) talks about the extent to which the interface and the system's underlying domain representation are actually independent: "the interface becomes epistemically removed from the system's actual domain expertise to the degree that it takes advantage of the student's ability to read more into its presentations than the system is aware of" (Wenger, *op. cit.*, p. 313). He contrasts the icons of Guidon-Watch (Richer and Clancey, 1985) which merely highlight internal processes, with those of Steamer (Hollan, Hutchins and Weitzman, 1984), which illustrate concepts not otherwise available to the system's internal model.

4.3 Implications for student modelling

Interfaces which are designed with a uniform and consistent metaphor can help in learning and using the system because they provide constraints within which the user can predict the effects of actions. The introduction of completely new potential (such as a "friendly" tutor intervening with advice) can break this closed system and lead to unpredictability. Hutchins (1987) provides suggestions for handling this tension between direct manipulation and declaration or conversational interaction. This "decoupling" of task-level and meta-level interaction is also advocated by Self (1988). However, the ability of a guidance system to collaborate with the user about the interaction presupposes that the system has some representation of the interface with which to reason and collaborate.

In this respect, discovery learning environments pose several problems for student modelling. Under the "model-world" metaphor, the learner's interaction with the system is far less constrained than in standard CAL or intelligent tutoring systems. Whilst simple and self-contained microworlds may be fairly easy to model, modelling the functionality of a much more general and powerful environment such as ARK is much more of a challenge. The problems are not simply to do with representing domain knowledge. The need to model the interaction, and have some notion of the sense that the user is making from interacting with the system, compound the problems. These issues are not specific to learning environments, but they become sharply focussed when the aim is to provide adaptive learning environments with some sort of guidance built into the system.

There are two senses in which one might understand the term "user modelling". One is the sense in which (cognitive) models of users are constructed in order to inform the design or evaluation of the system. Examples of such models are GOMS (Card, Moran & Newell, 1983) and TAG (Payne & Green, 1986). The other sense is closer to student modelling research — models of users are employed to provide intelligent, adaptive interfaces. Ideally, one would use the former kind of model in order to construct the latter kind. In practice, little attention is paid to the way in which users really operate.

Both types of approach have been applied with varying degrees of success to command-language interfaces (e.g., Unix), although the domain which the model can cover is usually restricted. Command-language interfaces are reasonably tractable, since the user is generally restricted to sequential (line-at-a-time) input representing discrete tasks. However, graphical interfaces, and systems that allow multi-tasking and background processing, are much more difficult.

There are two main sources of data for a system to model the student's progress and understanding. One source is from active diagnosis by the system, where the student is asked specific questions or required to solve particular problems generated by the system (e.g., IDEBUGGY — Burton, 1982; Pixie — Sleeman, 1987). The other source ("passive" diagnosis) is from the student's interaction with the system (e.g., Flow — Gentner, 1977; Anderson's geometry and Lisp tutors — Anderson, Boyle and Reiser, 1985). Some systems are designed such that the student's interaction is highly constrained, reducing the possibilities for ambiguity in diagnosis. The geometry tutor is one example, where the interface is designed so that it represents the steps of the problem solving task explicitly. This not only allows the student to reflect upon and learn from their own progress, but it also provides a reasonably unambiguous source of data about the student's reasoning processes. There is a tradeoff, however, in loss of flexibility of interaction. Other systems allow for a mixture of diagnostic strategy in interactive diagnosis. For example, Shopping-on-Mars

(Hennessy et al., 1987; Evertsz, Hennessy and Devi, 1988) uses a graphical interaction language for active diagnosis of students' arithmetic procedures, although the student is given some degree of flexibility in expressing his or her understanding.

Most current approaches to modelling the user's interaction with a system rely on a planning approach. That is, users are assumed to have prespecified goals which are translated into actions via some form of planning or problem solving. One source of problems is that users don't conform to the models of planning which AI systems use. Users activities are interleaved, they get interrupted, they suspend tasks. In order to keep track of complex interleaving plans, the system would have to be designed such that activity management was explicitly represented in the interface — an example of a system which supports this explicitly is Rooms (Card & Henderson, 1987), although the system is not designed for user modelling.

Another source of problems is that the planning approach assumes that the user has a definite goal in performing a task. But users' behaviour is not the result of purely internal goal-directed activity, as Suchman (1987) and others have pointed out. It involves the interaction between goal-directed activity and the actual physical and functional situation. This results in much more opportunistic, data-driven behaviour, and the development of partial plans or local tactics as much as more systematic problem solving. (See Elsom-Cook, 1989, for an attempt to reconcile Suchman's criticisms with planning models.)

The data-driven nature of interaction with graphical and direct manipulation interfaces implies that the system should be able to reason not only about its own state at any moment, but also about how that state appears to the user. That is, the system needs to have, in some sense, a representation of what's happening on the screen at any moment, as well as what's happening in terms of the domain being represented. Recently, there has been some attempt within ITS research to model the interaction in terms of the user-computer interface (cf. Spensley, 1988), however this is chiefly because the domain to be taught is the interface itself, so the issues of relating the interaction to the domain are still not tackled.

Plan-based approaches to modelling may result in the system assuming that the user (or learner) knows much more than they do in fact. Most HCI analyses of learnability make their predictions on the basis of information that is necessary (logically) to know. However, recent research suggests that users do not necessarily retain all the information they need to perform a task. Mayes, Draper, McGregor and Oatley (1988) showed that even experienced MacWrite users were able to recall very little of the visual details of the screen. They seemed to encode menus and menu items spatially, relying on this to find the items they needed, rather than encoding any details of which item was contained in which menu. This suggests that we don't need to encode explicitly (or learn in any strict

sense) the information that is required to perform tasks when the environment supports and encourages recognition. It suggests that knowledge of using a system is distributed between what's in the user's head and what's in the environment (cf. Norman, 1988).

Another problem is raised by systems that allow the user to construct their own microworlds (e.g., ARK). This is somewhat similar to the issue of modelling programming (e.g., in Logo, Lisp etc.). The difference is that in ARK the user can construct completely novel objects and functions from the basic building blocks. Whilst the system could model the units which comprise the new encapsulated entity, it couldn't model the emergent properties and semantics which this entity might have for the user.

If we take seriously the point that users activities are much more opportunistic than the goal-driven planning AI models suggest, then we need to worry about the issue of goal formulation. Current models of planning assume that goals are "givens". While some behaviour may be characterised as plan-driven, there are other occasions where it is only after users have acted and observed the outcome that they identify that state as being the goal to which the activity was (in retrospect) directed. This suggests that a model needs to be able to make inferences about what connections the user may have made between actions and their observed outcomes. It implies some explicit model of learning-by-doing. Some recent research (e.g., Lewis, 1988), is beginning to address this issue of how users make connections between actions and outcomes in building a model of the system.

The issue of interpreting the learner's behaviour and attempting to model the development of understanding is not just a problem for machines. Studies of classroom interactions by Newman, Griffin and Cole (1989) demonstrate that, very often, the understanding that children have of the point of a task or activity can be very different from the teacher's. However, Newman et al. point out that this is not necessarily problematic. The teacher behaves "as if" the child performs with the goal the teacher had in mind. Through interacting with the teacher in this way, the child eventually "appropriates" the goals of the activity, but there is no real sense in which the child's activities were directed towards that goal in the way in which user modelling approaches assume.

What these points imply for intelligent tutoring systems is that the tutor can never be certain about the understanding that the student has. This implication is echoed in Elsom-Cook's theory of "bounded user modelling" (Elsom-Cook, 1988) and in recent work by Self (1988) and Baker (in press), who argue for "approximate" theories of the student's understanding, and the modelling of the student's beliefs rather than his or her knowledge.

5. Summary and conclusions

It has been argued in this chapter that the design of an interface to an educational environment embodies implicitly a domain representation. If the tutoring component is to model successfully the learner's interaction with the system then that representation needs to be made explicit and accessible to the tutor. The process of understanding the way in which users interact with computers has two outcomes that can be used to advantage in guided discovery tutoring: one is that we can understand better how to design learnable and usable interfaces for teaching, the second is that we can start to formalise some principles that an intelligent tutoring system could use in modelling the learner-machine interaction.

It is not clear where to draw the boundaries of user-computer interaction. If the learner-machine interface includes the prior understanding that the user brings to the interaction (as Norman, 1986, and others have argued), it also includes the context in which the learner is using the system and which shapes the expectations of the learner. The teacher in the classroom plays an important role in providing goals for interacting with the learning technology. Newman et al. argue that, rather than having the system replace (simulate) the teacher, it should be designed to act as a tool that mediates between the teacher and the child, by providing a common ground upon which both teacher and child (or, rather children) can reflect, discuss and collaborate. The "shared world" or "collaborative manipulation" style of interaction needs to be extended, therefore, to include not just the learner and the tutoring system, but the human tutor as well.

References

- Anderson, J.R., Boyle, C.F. & Reiser, B.J. (1985) Intelligent tutoring systems. *Science*, 228 (4698), 456-462.
- Baecker, R.M. & Buxton, W. (1987) *Readings in Human-Computer Interaction*. Los Altos, CA: Morgan Kaufmann.
- Baker, M. (In press) Arguing with the tutor. In M.T. Elsom-Cook (Ed.) *Guided Discovery Tutoring*. London: Paul Chapman Publishers.
- Bannon, L. (1986) Helping users help each other. In D.A. Norman & S.W. Draper (Eds.) *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Brown, J.S. (1989) Toward a new epistemology for learning. In C. Frasson & J. Gauthier (Eds.) *Intelligent Tutoring Systems at the Crossroads of AI and Education*. Norwood, NJ: Ablex.
- Brown, D.E. & Clement, J. (1987) Overcoming misconceptions in mechanics: A comparison of two example-based strategies. Paper presented at the AERA meeting, Washington, D. C., 20-24 April.
- Burton, R.R. (1982) Diagnosing bugs in a simple procedural skill. In D.H. Sleeman & J.S. Brown (Eds.) *Intelligent Tutoring Systems*. New York: Academic Press.
- Buxton, W. (1986) There's more to interaction than meets the eye: Some issues in manual input. In D.A. Norman & S.W. Draper (Eds.) *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Card, S. & Henderson, A. (1987) A multiple, virtual-workspace interface to support user task switching. *Proceedings of the CHI+GI'87 Conference on Human Factors in Computing Systems and Graphics Interface*. New York: ACM.
- Card, S., Moran, T. & Newell, A. (1983) *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Carroll, J.M. & Mack, R.L. (1985) Metaphor, computing systems and active learning. *International Journal of Man-Machine Studies*, 22, 39-57.
- Clancey, W.J. (1983) GUIDON. *Journal of Computer-Based Instruction*, 10 (1), 8-14.

- Clement, J. (1988) Observed methods for generating analogies in scientific problem solving. *Cognitive Science*, 12, 563-568.
- Collins, A. & Brown, J.S. (1988) The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.) *Learning Issues for Intelligent Tutoring Systems*. New York: Springer-Verlag.
- DiSessa, A.A. (1986) Models of computation. In D.A. Norman & S.W. Draper (Eds.) *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Druin, A. (1988) NOOBIE: The animal design playstation. *SIGCHI Bulletin*, 20 (1), 45-53.
- Elsom-Cook, M.T. (1988) Guided discovery tutoring and machine learning. In J.A. Self (Ed.) *Artificial Intelligence and Human Learning*. London: Chapman and Hall.
- Elsom-Cook, M.T. (1989) Dialogue and teaching styles. In P. Goodyear (Ed.) *Intelligent Tutoring and Teaching Knowledge*. Chichester: Wiley.
- Elsom-Cook, M.T. (In press) Guided discovery tutoring. In M.T. Elsom-Cook (Ed.) *Guided Discovery Tutoring*. London: Paul Chapman Publishers.
- Evertsz, R., Hennessy, S. & Devi, R. (1988) *GADL: A Graphical Interface for Mental Arithmetic Algorithms*. Technical Report No. 49, Centre for Information Technology in Education, The Open University.
- Foley, J. (1987) Interfaces for advanced computing. *Scientific American*, 257 (4), 82-90.
- Frye, D. & Soloway, E. (1987) Interface design: A neglected issue in educational software. *Proceedings of the CHI+GI'87 Conference on Human Factors in Computing Systems and Graphics Interface*. New York: ACM.
- Gaver, W.W. (1986) Auditory icons: Using sound in computer interfaces. *Human-Computer Interaction*, 2, 167-177.
- Gaver, W.W. (1989) The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4, 67-94.

Gentner, D. R. (1977) The FLOW tutor: A schema-based tutorial system. *Proceedings of the Fifth Joint International Conference on Artificial Intelligence*. Cambridge, MA.

Gittins, D. (1986) Icon-based human-computer interaction. *International Journal of Man-Machine Studies*, 24, 519-543.

Hemenway, K. (1982) Psychological issues in the use of icons in command menus. *Proceedings of the Conference on Human Factors in Computing Systems* (Gaithersburg, Maryland). New York: ACM.

Hennessey, S., Evertsz, R., Ellis, D., Black, P., O'Shea, T. & Floyd, A. (1988) Design specification for 'Shopping on Mars', a computer-based educational activity. CITE Technical Report No. 29, Centre for Information Technology in Education, The Open University.

Hollan, J.D., Hutchins, E.L. & Weitzman, L. (1984) STEAMER: An interactive inspectable simulation-based training system. *AI Magazine*, 5 (2), 15-27.

Hutchins, E.L. (1987) *Metaphors for interface design*. ICS Report No. 8703, Institute for Cognitive Science, University of California, San Diego.

Hutchins, E.L., Hollan, J.D. & Norman, D.A. (1986) Direct manipulation interfaces. In D.A. Norman & S.W. Draper (Eds.) *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Johnson, P., Diaper, D. & Long, J. (1985) Task analysis in interactive system design and evaluation. In G. Johannsen, C. Mancini & L. Martensson (Eds.) *Analysis, Design and Evaluation of Man-Machine Systems*. Oxford: Pergamon Press.

Jones, S. (1983) Stereotyping in pictograms of abstract concepts. *Ergonomics*, 26, 605-611.

Krasner, G.E. & Pope, S.T. (1988) A cookbook of using the Model-View-Controller user interface paradigm. *Journal of Object-Oriented Programming*, 1 (3), 26-49

Lansdale, M.W. (1988) On the memorability of icons in an information retrieval task. *Behaviour and Information Technology*, 7, 131-151.

Lewis, C. (1988) Why and how to learn why: Analysis-based generalization of procedures. *Cognitive Science*, 12 (2), 211-256.

Mayes, J.T., Draper, S.W., McGregor, A.M. & Oatley, K. (1988) Information flow in a user interface: The effect of experience and context on the recall of MacWrite screens. In D.M. Jones & R. Winder (Eds.) *People and Computers IV*. Cambridge: Cambridge University Press.

Muller, M.J. (1988) Multifunctional cursor for direct manipulation user interfaces. *Proceedings of the CHI'88 Conference on Human Factors in Computing Systems*. New York: ACM.

Newman, D., Griffin, P. & Cole, M. (1989) *The Construction Zone: Working for Cognitive Change in School*. Cambridge: Cambridge University Press.

Norman, D.A. (1986) Cognitive engineering. In D.A. Norman & S.W. Draper (Eds.) *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D.A. (1988) *The Psychology of Everyday Things*. New York: Basic Books.

O'Malley, C. (1987) Understanding explanation. *Proceedings of the Third Alvey Workshop on Explanation*. London: IEE.

O'Shea, T., O'Malley, C. & Scanlon, E. (1988) *Magnets, martians and microworlds: Learning with and learning by OOPS*. CITE Technical Report No. 53, Centre for Information Technology in Education, The Open University.

O'Shea, T. & Smith, R. (1987) Understanding physics by violating the laws of nature: experiments with the Alternate Reality Kit. *Proceedings of the Conference on Computer-Assisted Learning (CAL'87)*, University of Strathclyde.

Pailhous, J. & Bullinger, A. (1978) The role of interiorisation of material properties of information-acquiring devices in exploratory activities. *Communication and Cognition*, 11, 209-234.

Payne, S.J. & Green, T.R.G. (1986) Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction*, 2 (2), 93-133.

Richer, M.H. & Clancey, W.J. (1985) GUIDON-WATCH: A graphic interface for viewing a knowledge-based system. *IEEE Computer Graphics and Applications*, 5 (11), 51-64.

Riley, M. & O'Malley, C. (1985) Planning nets: A framework for analyzing user-computer interactions. In B. Shackel (Ed.) *Human-Computer Interaction — Interact'84*. Amsterdam: North-Holland.

Rogers, Y. (1988) Pictorial representations of abstract concepts in relation to human-computer interaction. Unpublished PhD thesis, University of Wales.

Rogers, Y. (1989) Icons at the interface: Their usefulness. *Interacting with Computers*, 1 (2), 105-117.

Rohr, G. & Keppel, E. (1984) Iconic interfaces: Where to use and how to construct. *Human Factors in Organisational Design and Management*. Amsterdam: North-Holland.

Scaife, M. (1987) The need for developmental theories in cognitive science: Children and computing systems. In J.C. Rutkowska & C. Crook (Eds.) *Computers, Cognition and Development*. Chichester: Wiley.

Scanlon, E. & O'Shea, T. (1988) Cognitive economy in physics reasoning: Implications for designing instructional materials. In H. Mandl & A. Lesgold (Eds.) *Learning Issues for Intelligent Tutoring Systems*. New York: Springer-Verlag.

Scanlon, E. & Smith, R. (1988) A rational reconstruction of a bubble chamber simulation using the Alternate Reality Kit. *Computers & Education*, 12 (1), 199-207.

Self, J. (1988) Bypassing the intractable problem of student modelling. *Proceedings of the 1988 Conference on Intelligent Tutoring Systems*, Montréal, 1-3 June.

Shneiderman, B., (1983) Direct Manipulation: A step beyond programming languages. *IEEE Computer*, 16 (8), 57-63.

Shrager, J. (1987) Theory change via view application in instructionless learning. *Machine Learning*, 2, 1-30.

Sleeman, D.H. (1982) Assessing aspects of competence in basic algebra. In D.H. Sleeman & J.S. Brown (Eds.) *Intelligent Tutoring Systems*. New York: Academic Press.

Smith, R.B. (1986) The Alternate Reality Kit: An animated environment for creating interactive simulations. *Proceedings of the IEEE Workshop on Visual Languages*. Washington: IEEE.

Smith, R.B. (1987) Experiences with the Alternate Reality Kit: An example of the tension between literalism and magic. *Proceedings of the Conference on Human Factors in Computing Systems (CHI+GI'87)*. New York: ACM.

Spensley, F.M. (1988) *DOMINIE: Trainer interface*. CITE Technical Report No. 44, Centre for Information Technology in Education, The Open University.

Suchman, L.A. (1987) *Plans and Situated Actions*. Cambridge: Cambridge University Press.

Szlichcinski, K.P. (1980) The product and comprehension of pictorial instructions. Unpublished PhD thesis, University of London.

Weitzman, L. (1986) *Designer: A knowledge-based graphic design assistant*. ICS Report no. 8609, Institute for Cognitive Science, University of California, San Diego.

Wenger, E. (1988) *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.

White, B. & Fredriksen, J. (1987) *Causal model progressions as a foundation for intelligent learning environments*. Bolt, Beranek and Newman Technical Report No. 6686, Boston, MA.

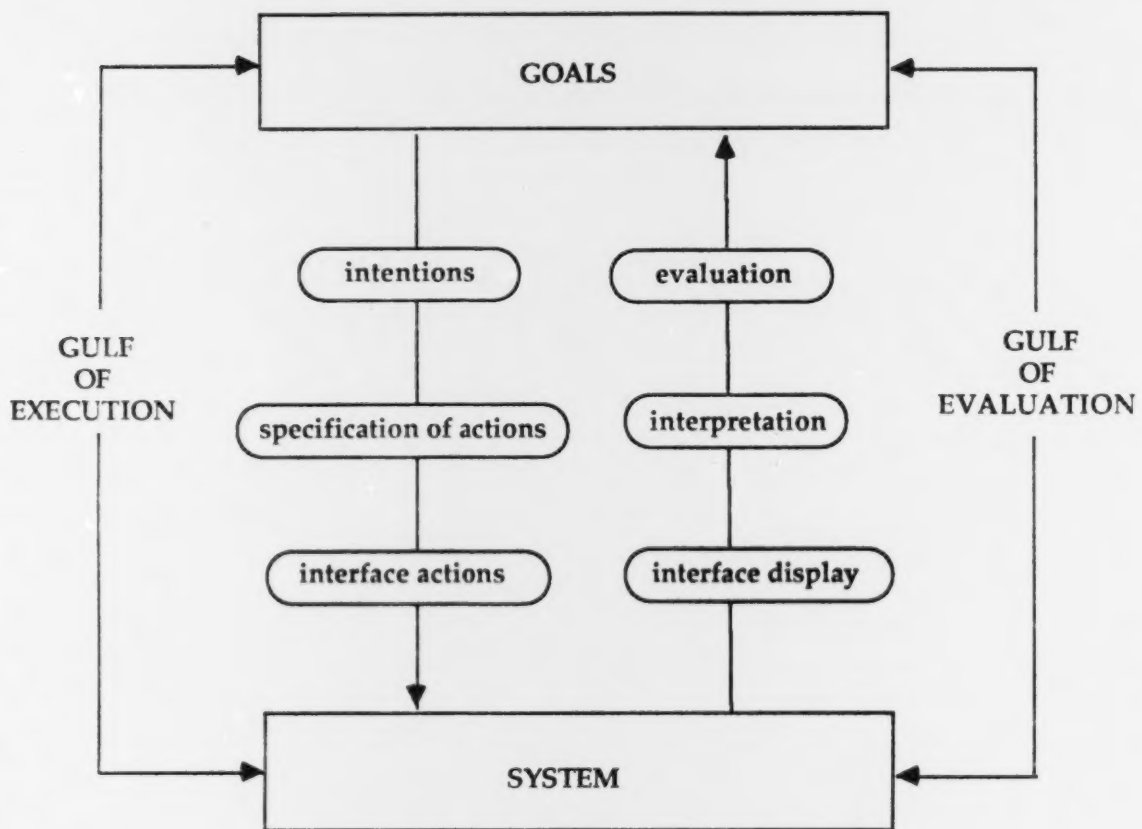


Figure 2.1 The gulf of execution is bridged by the user in forming intentions and the determining an action sequence. It is bridged by the system when the designer builds the input characteristics of the interface. The gulf of evaluation is bridged by the user in perceiving the system state, interpreting it and evaluating it with respect to some set of goals. It is bridged by the system when the designer builds the output characteristics of the interface (Norman, 1986).



(a) Icon representing a document in the Apple Macintosh interface.



(b) Icons representing documents created with three different word processing applications (Microsoft Word, MacWrite and WriteNow).



(c) Icons representing graphics documents (MacDraw, MacPaint and Canvas).

Figure 2.2 Factors affecting the identification of referents from icons: (a) visual link between icon and referent (in this case, a document); (b) the extent to which referents can be differentiated within a set of icons (in this case the document icons refer to different word processing applications); (c) common characteristics of referents (these icons all represent graphics documents).

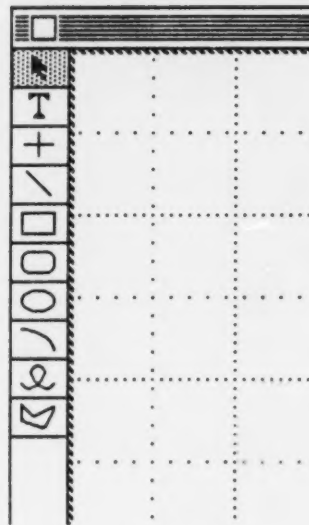


Figure 2.3 The tools menu in MacDraw. The fact that this is a graphics application constrains the interpretation of each of these icons — the shapes represent the kinds of things the user can draw (e.g., rectangles, curves, polygons, etc.)



(a) Symbolic mapping representing no entry.



(b) Nomic mapping representing a folder.

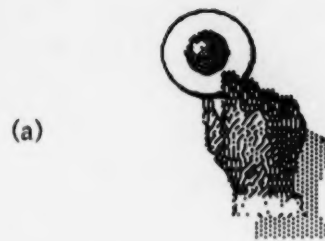


(c) Metaphorical mapping where structural relations represent an organisation such as a hierarchy.



(d) Metaphorical mapping using metonymy, where the receiver represents a telephone.

Figure 2.4 Types of mapping between icons and referents.



mouse enters



mouse down

Figure 2.5 Cursor feedback in pressing buttons.

(a)



mouse up

(b)



mouse down

(c)



mouse still down

(d)



mouse still down
& drag

Figure 2.6 Cursor feedback in moving objects.

CENTRE FOR INFORMATION TECHNOLOGY IN EDUCATION

List of CITE Reports

These reports may be obtained from:

Hansa Solanki, Institute of Educational Technology, The Open University, Walton Hall,
MILTON KEYNES, MK7 6AA, England.

<u>Report No.</u>	<u>Title and Author</u>
1	A.T. Vincent, (1985) Computing and the Blind.
2	A. Jones, G. Kirkup, J. Morrison (1985) A Trial of Home Based Computer Terminals.
3	Gill Kirkup, (1985) The Present and Potential Use of Ceefax in the Open University.
4	Mark Elsom-Cook, (1986) Artificial Intelligence and Computer Assisted Instruction.
5	Mark Elsom-Cook, (1986) A Pascal program checker.
6	Simon Holland, (1986) How computers are used in the teaching of music and speculations about how Artificial Intelligence could be applied to radically improve the learning of composition skills.
7	Simon Holland, (1986) Design consideration for a human-computer interface using 12-tone three-dimensional harmony space to aid novices to learn aspects of harmony and composition.
8	Alison Petrie-Brown, (1987) The Influence of Context and Coherence as a Foundation for Dialogue Research.
9	Eileen Scanlon, Randall B. Smith (1987) A Rational Reconstruction of a Bubble Chamber Simulation Using The Alternate Reality Kit.
10	Mark Elsom-Cook, (1987) Intelligent Computer-Aided Instruction research at the Open University.
11	Mark Elsom-Cook, (1987) Towards a framework for human-computer discourse.
12	Mark Elsom-Cook, (1987) MATILDA AND IMPART: Lisp tools.
13	Mark Elsom-Cook, (1987) Guided discovery tutoring and bounded user modelling in Intelligent Computer Aided Instruction ed. J. Self, Chapman-Hall 1987.
14	A.M. Petrie-Brown and M.T. Elsom-Cook, (1987) An Examination of an AI model of indirect speech acts.
15	A. Edwards, (1987) Integrating Synthetic Speech With Other Auditory Cues In Graphical Computer Programs For Blind Users.
16	S. Holland, (June 1987) A knowledge-based tutor for music composition.
17	S. Holland, (June 1987) New Cognitive Theories of Harmony Applied To Direct Manipulation Tools for Novices.

- 18 M. Baker. (July 1987) Intelligent Computer-Aided Instruction and Musical Performance Skills.
- 19 M. Baker, (August 1987) Proposed Research Directions for Intelligent Computer-aided Instruction in Musical Performance Skills.
- 20 A D N Edwards, (August 1987), Adapting interfaces for visually disabled users.
- 21 M Elsom-Cook, (September 1987), Acquisition of computing skills.
- 22 M. Baker, (September 1987), Computational Analysis of musical grouping structures.
- 23 M Baker, (June 1987), Automated Analysis of Musical Grouping Structures as a Basis for a Guided Discovery Environment for Interpretation of Music.
- 24 D Laurillard, (October 1987), The different forms of learning in psychology and education.
- 25 A.D.N. Edwards, (November 1987) Modelling blind users" interactions with auditory computer interface.
- 26 P. Fung, (November 1987) Novice Prolog Programmers.
- 27 P. Fung, B. DuBoulay & M. Elsom-Cook, (November 1987), An initial taxonomy of novices' misconceptions of the Prolog interpreter.
- 28 G. Kirkup, (November 1987), Considering the effect on women students of an increased use of microcomputers in distance education.
- 29 Sara Hennessy, Rick Evertsz, Dave Ellis, Phil Black, Tim O'Shea, Ann Floyd, Design Specification for 'Shopping on Mars' a computer-based Educational Activity.
- 30 A D N Edwards, The Use of home computers by disabled students at the Open University. Part 1: Previous use of computers in courses
- 31 Michael E. Fox (March 1988) A Report on Studies of Motivation, Teaching and Small Group Interaction with special reference to Computers and to the Teaching and Learning of Arithmetic.
- 32 Michael E. Fox (March 1988) Theory and Design for a Visual Calculator for Arithmetic.
- 33 M. Baker (February 1988) A Cognitive Model for Perception of Musical Grouping Structures
- 34 M. Baker (February 1988) An Artificial Intelligence Approach to Musical Grouping Analysis
- 35 P. Fung (December 1987) Novices' predictions of Prolog's control flow: A report on an empirical study
- 36 M. Elsom-Cook and F. Spensley (April 1988) Knowledge representation in a tutoring system for procedural skills.
- 37 F. Spensley & M. Elsom-Cook (April 1988) Dominic: Teaching and Assessment Strategies.
- 38 M. Baker (May 1988) Tutoring with Incomplete and Uncertain Knowledge
- 39 R. Moyse (May 1988) Multiple Viewpoints for Intelligent Tutoring Systems.

- 40 P. Whalley (May 1988) Hyper Technic - a graphic object-oriented control environment.
- 41 P. Whalley (June 1988) Cued recall as a Measure of argument integration.
- 42 L. Alpay (June 1988) A Survey and Examination of Intelligent Tutoring Systems in Medicine
- 43 M. Elsom-Cook (July 1988) Introduction to the ECAL system.
- 44 F. Spensley (July 1988) Dominie: Trainer Interface.
- 45 M. Baker (July 1988) An Architecture of an Intelligent Tutoring System for Musical Structure and Interpretation.
- 46 E. Scanlon and R. Sibbitt (April 1988) Proceedings of CAL Conference 88.
- 47 L. Alpay (September 1988) Medical Problem-Solving and Intelligent Tutoring Systems: Proposed Research Directions.
- 48 S.A Cerri, M.T. Elsom-Cook, M. Leoncini (September 1988) TRILL: The Rather Intelligent Little Lisper.
- 49 R. Evertsz, S. Hennessy & R. Devi (September 1988) GADL: a Graphical Interface for Mental Arithmetic Algorithms.
- 50 P. Fung, (October 1988) A formalisation of novices' errors in Prolog programs.
- 51 M. Baker, (October 1988) Arguing with the tutor: a model for tutorial dialogue in uncertain knowledge domains.
- 52 (October 1988) Information Technology in Education: Conceptual change in Science.
- 53 T. O'Shea, C. O'Malley & E. Scanlon (October 1988) Magnets, Martians and Microworlds: Learning with and Learning by OOPS.
- 54 (November 1988) Directory of Research 1988.
- 55 P. Fung (November 1988) Automated Diagnosis of Prolog control flow errors: a first evaluation.
- 56 R. Mason (December 1988) The use of Computer-Mediated Communication for Distance Education at the Open University, 1988.
- 57 A.D.N. Edwards & P. Grove (December 1988) A User's Guide to the Voice Screen Reader.
- 58 R. Sibbitt (December 1988) How Children Spell Nonsense Words.
- 59 Directory of Research 1988-1989.
- 60 R. Joiner (February 1989) Mechanisms of Cognitive Change in Peer Interaction: A Critical Review.
- 61 R. Devi (February 1989) Machine Learning and Tutoring System.
- 62 M. Elsom-Cook (February 1989) Dialogue and teaching styles.
- 63 R. Moyse (March 1989) Knowledge Negotiation Implies Multiple Viewpoints.

- 64 D. Laurillard (March 1989) University of Hong Kong Medical Education Symposium. December 1-3 1988. Understanding Medical Students' Problem-Solving.
- 65 A. Kirkwood and G. Kirkup (March 1989). Computing on DT200, M205 & M371 - Report of the Initial Survey of Spring 1988.
- 66 R. Evertsz (April 1989) Two papers on the Abstract Interpretation of Production Systems.
- 67 M. Elsom-Cook and C. O'Malley (April 1989) ECAL: Bridging the gap between CAL and ITS¹.
- 68 D. Laurillard (May 1989) CAL and Numeracy.
- 69 G. Kirkup and E. Dale (May 1989) Home Computing Evaluation Report. M205 End of Year Report 1988.
- 70 A. Jones (May 1989) Home Computing Evaluation Project. M371 End of Year Report 1988.
- 71 A. Blandford (May 1989) Engineering Design Education and Intelligent Tutoring Systems.
- 72 D Hawkrige (May 1989) Towards 2000: Informatics in Education.
- 73 D. Hawkrige (May 1989) Who Needs Computers in Schools, and Why?
- 74 C. O'Malley, M. Elsom-Cook and E. Ridwan (May 1989) The ECAL authoring environment.
- 75 C. O'Malley and E. Scanlon (May 1989) Computer-supported collaborative learning: Problem solving and distance education.
- 76 M. Thorpe (June 1989) Home Computing Evaluation Project. The Tutor Perspective on Computer Mediated Communication in DT200: Introduction to Information Technology.
- 77 A. Kirkwood and E. Dale (May 1989) Home Computing Evaluation Project. DT200 End of Year Report 1988.
- 78 R. Hewson (June 1989) Investigating the circumstances of a specific document.
- 79 R. Hewson (June 1989) Typographic Design: One Practitioner's View of Old Meets New.
- 80 G. Kirkup (July 1989) Home Computing Evaluation Project. T102 Set-up, Start-up and Block one. Report on February Survey 1989.
- 81 L. Alpay (August 1989) Acquisition of Reasoning Strategies in Medical Diagnosis.
- 82 D. Laurillard (August 1989) T102 CAL Numeracy. Formative Evaluation Report, July-August 1989.
- 83 G. Kirkup (August 1989) Home Computing in Practice 1988-1989 and Possibilities for 1992.
- 84 C. Saxton (August 1989) Home Computing Evaluation 1988. To Compute or Not to Compute.

